LA-SIGMA REU-SUBR

Stereographic Visualization of Molecular Configurations in a CAVE

G'Nita R. Wright[1], Sanjay Kodiyalam[2], Amitava Jana[2]
Department of [1]Biological Sciences / [2]Mechanical Engineering
Southern University, Baton Rouge, Louisiana.

7/25/2011

Abstract

Motivated by the need for visualizing large scale molecular configurations from parallel atomistic simulations we have developed a Visual Studio C/C++ project using the CAVE and GLU libraries for stereographically visualizing atoms and bonds in a molecular configuration. In the absence of any simulation data the project constructs Bravais lattice configurations for visualization: the Simple Cubic lattice with only atoms and the Body Centered Cubic lattice with atoms and bonds. OpenGL lighting effects are used to enhance depth perception and to distinguish atom types. The CPU time required for the display versus the total number of atoms is determined. This timing identifies future tasks for improving the code's execution efficiency to enable display of million-atom configurations. The project may be used in education on visualization.

Introduction

Current research in the area of computational sciences focuses on virtual reality systems with emphasis on the Cave Automatic Virtual Environment (CAVE) for the visual inspection of the computed results. This work, on visualizing atomistic configurations, was conducted at the CAVE at Southern's College of Engineering (CoE) (Fig. 1a). This CAVE is an 8 ft x 8 ft x 8 ft space with 4 displays (3 on screen-walls and one on the floor). All displays are via mirrors used to set the required optical distance (projector's throw) within the limited space of the room housing the CAVE. Active stereo viewing is enabled by projecting rapidly alternating left- and right- eye images that are synchronized with the shutters of the user's stereo eye-ware. The position and orientation of two sensors are continuously tracked in the CAVE. They are attached to the user's eye-ware and wand. In order to perceive a 3-dimensional object the displays must change in accordance with the user's point of view. The required perspective transformations are automatically carried out by the CAVE library [1] with the visualization application developer needing to code for a single (non-stereo) display. This library also allows the application access to all the sensors' information which may then be used for interacting with / navigation through the virtual world by appropriate updates to the display. The CAVE is run by a two-node cluster: The master node collects the sensors' information while the display node drives the projectors.

While perspective transformations allow for depth perception even in a non-stereo display, the CAVE with stereo displays and user navigation enables clearer depth perception and exploration of the virtual world at all relevant length scales. This makes it an ideal environment for visualizing large scale molecular configurations from parallel atomistic simulations. The visualization will then motivate, and can be coupled to, subsequent analysis of the configurations. We have therefore begun the construction of such a visualization + analysis program with the eventual goal of being able to visualize ~million-atom configurations. A Visual Studio C/C++ project for visualization of atoms and bonds in a molecular dynamics configuration is developed and its performance (CPU time taken versus number of atoms) is determined. The CPU-timing is carried out on the development platform: a desktop with a non-stereo "CAVE Simulator" display.

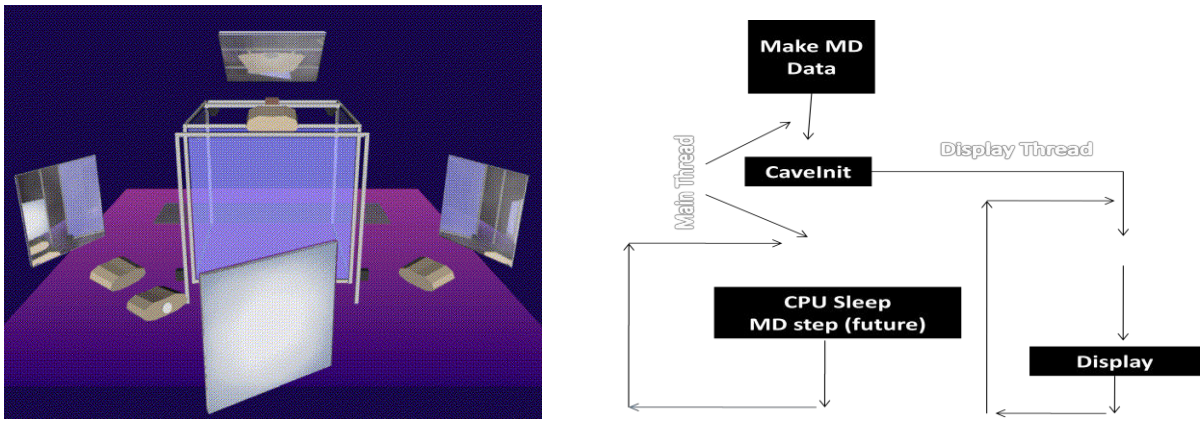

Figure 1. (a) Schematic [2] of the CAVE at Southern's CoE showing the projectors, mirrrors, and screens. (b) Threads involved in the operation of the CAVE library [1].

Methodology

The operation of the CAVE library (Fig. 1b) dictated the project's structure. The CAVE library automatically spawns synchronized threads for the displays in addition to the main thread that continues to operate asynchronously under user control. The display threads are not under user control but simply call a user defined initialization function once followed by an infinite loop that has calls to the user defined navigation and display functions.

Within the main thread, in the absence of data from any atomistic simulation, a function (MakeMDdata) is implemented. This function is called once at the beginning of project execution. It creates arrays corresponding to the type (an integer), position & velocity (3 double precision numbers each) for the set of atoms to be visualized. The positions of the atoms are corresponding to either a Simple Cubic or Body Centered Cubic (BCC) Bravais lattice that are both spatially periodic structures [3]. While the former has only one type of atom the latter has two types with the atoms at the cube-corners distinguished from those at the cube-center. The velocities are set to zero as they are not subsequently used. The velocity array is nevertheless provided as both position and velocity together completely define the state of an atomistic configuration – for future use when reading in data from a simulation. The main thread does not have any other functions – it is kept from termination by an infinite loop that allows for the CPU to sleep.

Using standard OpenGL an initialization function within the display thread (InitDisplay) is used for setting up the lighting that remains fixed for the entire execution. A single directional light, with ambient, diffuse, and specular components, is used in an attempt to enhance depth perception. The ambient component corresponds to light being reflected equally in all directions by the visualized surfaces, the diffuse & specular components correspond to reflection in broad and narrow angular ranges around the ideal mirror-like reflection. The emissive component of lighting corresponding to light emitted by surfaces independent of an incident light was not used.

The navigation function within the display thread is adopted from earlier work [4]. It enables the translation, rotation, and scaling of the entire virtual world using the wand. The translation occurs in the direction pointed to by the front of the wand which also serves as the axis of rotation. As all the navigation operations are in the CAVE's coordinate system the rotation and scaling are centered around the origin of the CAVE: The mid-point of the floor-display.

The display function uses standard OpenGL commands and the GLU library. Atoms are displayed within a loop spanning the total number of atoms. They are displayed as spheres using the function gluSphere from the GLU library. This function displays a sphere, at the current OpenGL origin, with a user-defined radius and surface smoothness. The position of each atom is therefore set using the OpenGL transformation function glTranslated before the call to gluSphere. For the independent action of transformations corresponding to different atoms the display of each atom is enveloped by a glPushMatrix-glPopMatrix pair. The smoothness of the atom-sphere is controlled by two integer parameters: the slices (longitudes) and stacks (latitudes) that the sphere is divided into for the purpose of its display: Via the display of the OpenGL primitives - triangles or quadrilaterals defined by the intersections of neighboring latitudes & longitudes. We therefore assume the measure of smoothness to be the product of the two parameters: Smoothness measure = numbers of slices x numbers of stacks. Different atom types are distinguished by their color that is set using the glColorMaterial function. While the specular reflection quality is set to a constant white color (and the shinyness ficed at 1.0), the ambient and diffuse components are set to green or blue color. Bonds are displayed as red lines, setting GL_LINES, between atoms of different type – and therefore exist only in the BCC lattice configuration. Only "nearest neighbor" atoms are bonded [5]. In order to determine the nearest neighbors of a particular atom the $(\text{distance})^2$ to *all other atoms* of different type is determined: Bonds are drawn only between those atom pairs for which this $(\text{distance})^2$ is less than the expected $(\text{bondlength})^2 \times 1.01$. The display of bonds therefore requires a "double loop" – a (first) loop over all atoms enveloping another (second) loop over all the atoms.

The performance of the code is studied by determining the CPU time required by the display function for displaying either all the atoms or all the bonds once (= one frame). The separation of atoms and bonds is in the expectation that the trend in the variation of this time per frame, with increasing total number of atoms, is different for the two cases. This time is determined as an average over many frames – typically 100. The increase in this time with increasing total number of atoms then indicates the limiting number of atoms the project can handle while having an *interactive* frame rate of ~10 frames/second *i.e.* when the time per frame reaches 0.1 second. This timing study is motivated by the eventual goal of having an efficient program capable of handling ~million atom configurations.

Results

While the project executes as expected on the machines driving the CAVE it is tested on the development platform producing a single display in non-stereo "CAVE simulator" mode that is a perspective view of the entire CAVE (Fig. 2). This platform is a Windows-XP desktop having 2 GB of RAM and 4 CPU (2 x dual core Opterons, 2.4 GHz).

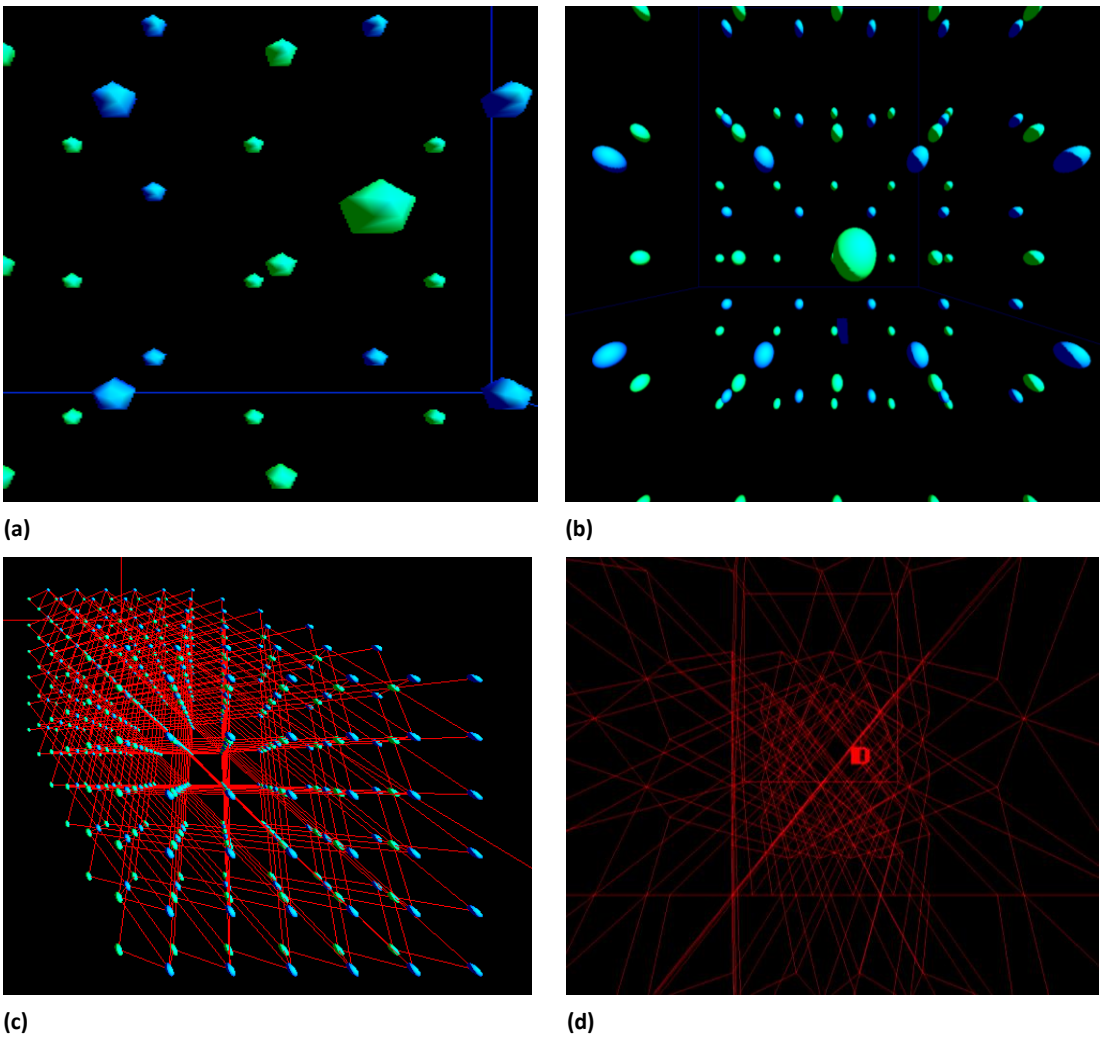

Figure 2. Perspective views of atoms / bonds in the BCC lattice configurations from the CAVE simulator display on the desktop. (a) shows only the atoms as coarse spheres – each composed out of 5 slices and 5 stacks. (b) shows only the atoms as smooth spheres – each composed out of 50 slices and 50 stacks. (c) shows both atoms and bonds. (d) shows only the bonds.

Figure 2 shows images of the BCC lattice configuration. The first atom type is shown in green, the second in blue, and the bonds in red. When the sphere-atom is composed of only 5 slices and 5 stacks its display is rather coarse (Fig. 2a). This sphere becomes smooth when the number of slices and stacks are both increased to 50 (Fig. 2b). Each atom inside the lattice has eight bonds as expected (Fig. 3c). Images with only bonds (Fig. 3d) are generated for timing purposes.

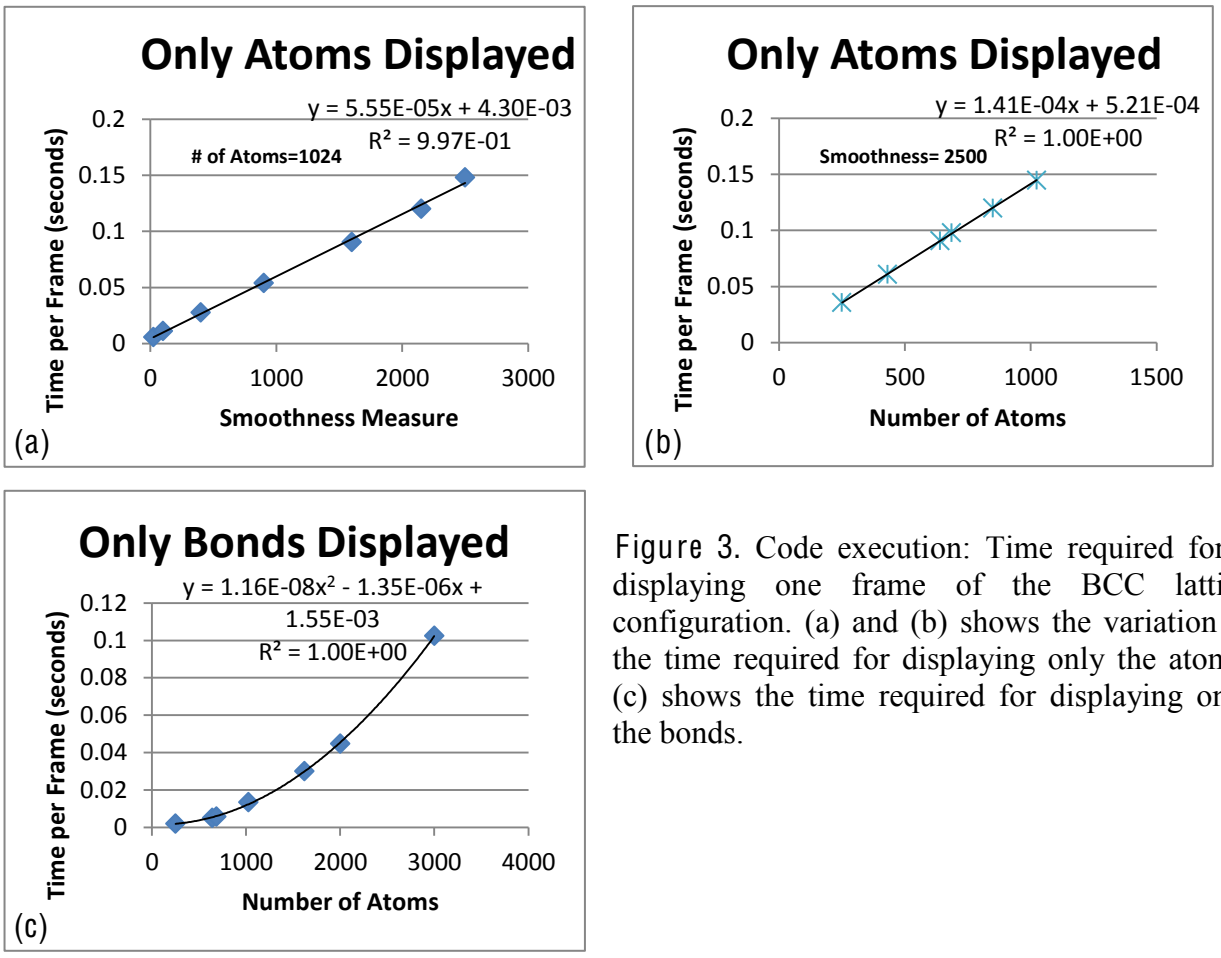

Figure 3. Code execution: Time required for a displaying one frame of the BCC lattice configuration. (a) and (b) shows the variation in the time required for displaying only the atoms. (c) shows the time required for displaying only the bonds.

Figure 3 shows the variation in the time required to display one frame (T) with only atoms or only bonds. Figs. 3a & 3b show that with atoms alone this time is proportional to both the number of atoms and the smoothness measure. This behavior is as expected, but was not demonstrated, in earlier work [6]: the time is proportional to the total number of Open GL primitives (triangles / quadrilaterals) that need to be displayed. The time required for displaying bonds alone (Fig. 3c) is seen to vary roughly as the (Number of atoms)^2. It is understood as being due to the double loop over atoms needed for the determination of neighboring atoms. From the curve fits to the timing graphs (Figs. 2b & c) the upper bound to the number of atoms that the project can currently handle with an interactive frame rate, *i.e.* with T = 0.1 seconds, is 706 and 2913 respectively for the display of atoms alone or bonds alone respectively.

Conclusion

In the context of developing a coupled visualization and analysis program for studying data from parallel atomistic simulations we have developed a Visual Studio C/C++ project using the CAVE and GLU libraries to stereographically visualize atoms and bonds in a molecular

configuration. While data from any simulation has not been used molecular configurations in Simple Cubic and Body Centered Cubic lattice configurations have been constructed and visualized. The CPU time required for the display of atoms alone scales linearly with the total number of atoms where as for the display of bonds alone the scaling is quadratic in the total number of atoms. This scaling is understood as being due to the single and double loops involved in the display of atoms and bonds respectively. The performance of the current version of the project leads to an upper bound in the number of atoms that can be handled with an interactive frame rate of 10 frames per second. This bound is 706 (2913) for the display of atoms (bonds) alone.

The current limitations of the project identify tasks for improving the code's execution efficiency to eventually enable the interactive display of million-atom configurations such as those from molecular dynamic simulations of protein-ligand docking. Display rate of atoms can be increased by having a variable smoothness measure for the sphere-atom depending on the perspective angular width of the atom as has been implemented in earlier work [7]. Bond display can be made to scale linearly with the number of atoms by using linked and neighbor lists as has been stated, but not demonstrated, in earlier work [6]. A more robust display of the bond using cylinders can be attempted.

This project, in its current version and its future development, can be used in education on visualization. As the algorithms of use in this project are applicable widely – including molecular dynamics simulations [6] – it may also be used in education on molecular dynamics simulation & analysis.

Acknowledgments

This work was funded by the Louisiana Board of Regents, through LASIGMA [Award Nos. EPS-1003897, NSF (2010-15)-RII-SUBR, and HRD-1002541]. One of the authors (G. R. Wright) thanks Dr. Diola Bagayoko for the opportunity to conduct research with the LA-SiGMA REU.

References

[1] Dave Pape, Caoline Cruz-Neira, Marek Czernuszenko, CAVE Library version 2.6, Electronic Visualization Laboratory, University of Illinois at Chicago, (1997). Currently used: Version 3.2, CaveLib™, Mechdyne Corporation, Marshalltown, Iowa.

[2] Figure from http.cs.uic.edu/~kenyon/conference/GILKY/CAVE_DOD.html

[3] Mehta, S.; Hazzard, K.; Machiraju, R.; Parthasarathy, S.; Wilkins, J.; , "Detection and visualization of anomalous structures in molecular dynamics simulation data," *Visualization, 2004. IEEE* , vol., no., pp. 465- 472, 10-15 Oct. 2004.

[4] Ghadersohi, Amin. Center for Computational Research, SUNY-Buffalo. *Collaborative Scientific Visualization and Real-time Monitoring of Protein Structure Data.*

[4] D. Driggs, S. Kodiyalam, K., A. Jana, "Modeling Virtual Cooperative Robots," Proceedings of the 2011 ASEE-GSW Conference, March 2011, Houston, Louisiana, Session FC1-1.
[5] Ghadersohi, Amin. Center for Computational Research, SUNY-Buffalo. *Collaborative Scientific Visualization and Real-time Monitoring of Protein Structure Data*.
[6] S. Kodiyalam, A. Jana, "Computation and Stereographic Visualization of Molecular Dynamics in Undergraduate Education," Proceedings of the 2011 ASEE-GSW Conference, March 2011, Houston, Louisiana, Session T3B-3.
[7] A. Sharma, A. Nakano, R. K. Kalia, P. Vashishta, S. Kodiyalam, P. Miller, W. Zhao, X. Liu, T. J. Campbell, and A. Hass, *"Immersive and Interactive Exploration of Billion-Atom Systems,"* Presence: Teleoperators and Virtual Environments 12, 85 (2003).